RFC 7480 : HTTP usage in the Registration Data Access Protocol (RDAP)

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 26 mars 2015

Date de publication du RFC : Mars 2015

http://www.bortzmeyer.org/7480.html

Le protocole d'accès aux informations des registres RDAP <http://www.bortzmeyer.org/weirds-rdap.html> (*"Registration Data Access Protocol"*), qui se veut successeur de whois, peut utiliser plusieurs protocoles de transport différents. Pour l'instant, le seul normalisé est HTTP, dans ce RFC.

HTTP est normalisé dans le RFC 7230[1]. Mais, à part cela, il n'y a évidemment pas besoin de le présenter. Son choix comme protocole de transport va de soi dans l'Internet aujourd'hui, et permet de doter RDAP <http://www.bortzmeyer.org/weirds-rdap.html> d'une jolie sémantique REST. Toute la complexité est dans le serveur, un des objectifs de ce RFC est que « on puisse utiliser comme client RDAP des outils généralistes comme bash ou wget ». Le mode d'emploi est à peu près :

- Trouver le bon serveur, par exemple en utilisant la technique du RFC 7484,
- Fabriquer une jolie requête HTTP de type GET (RFC 7231), en suivant le RFC 7482, par exemple, pour trouver de l'information sur le réseau 192.0.2.0, ce sera http://example.net/rdap/ip/192.0.2.0,
- Envoyer la requête et lire la réponse en JSON (RFC 7483),
- La réponse était peut-être sur un autre serveur et, dans ce cas, on a reçu non pas le code JSON, avec le statut 200, mais une redirection (statut HTTP 3xx) ; dans ce cas, on réessaie avec l'URL indiqué dans la redirection.

D'autres réponses étaient possibles, comme le fameux 404 si on demande des informations sur un objet qui n'existe pas. RDAP utilise HTTP et on a donc toutes les subtilités de HTTP à sa disposition.

La section 3 de notre RFC résume les principes de RDAP-sur-HTTP :

- Une seule requête HTTP pour avoir l'information,
- Plusieurs formats de sortie possible en théorie même si, pour l'instant, seul le JSON du RFC 7483 est spécifié,

1. Pour voir le RFC de numéro NNN, http://www.ietf.org/rfc/rfcNNN.txt, par exemple http://www.ietf.org/rfc/rfc7230.txt

— Possibilité d'utiliser tout ce que permet HTTP, afin d'éviter de réinventer la roue. Ainsi, si on veut de la compression, on ne la réinvente pas dans RDAP, on utilise celle de HTTP (RFC 7231, section 3.1.2). Même chose pour les mécanismes d'authentification, où on suivra le RFC 7235 ou bien le cache, selon le RFC 7234. Ainsi, on pourra récupérer tous les logiciels et l'expertise de HTTP.

La section 4 détaille la formation des requêtes RDAP sur HTTP. Ce sont des requêtes `GET` ou `HEAD` (puisqu'elles ne modifient pas les données, RDAP étant un protocole en lecture seule), qui fonctionnent sur HTTP ou HTTPS (qui est obligatoire pour les logiciels RDAP). Le client doit penser à mettre un entête `Accept:` qui indique un type de données qui existe pour RDAP, comme le `application/rdap+json` du RFC 7483 (notez la syntaxe avec un +, issue du RFC 6839). S'il l'oublie, le serveur est libre de faire ce qu'il veut, le RFC suggérant d'envoyer une réponse acceptable par un navigateur Web ordinaire (comme `text/html`), car c'est sans doute cela qui provoquera le moins de surprise (les serveurs RDAP existants - voir par exemple `<http://rdg.afilias.info/rdap/domain/reflets.info>` - ne suivent pas cette recommandation).

La requête peut contenir des paramètres (après le ? dans l'URL).

La section 5 de notre RFC détaille les réponses possibles. Comme on utilise HTTP comme transport, tous les codes de réponse HTTP (RFC 7231, section 6) sont admissibles. Néanmoins, certains seront sans doute plus courants :

— 200 : réponse positive, tout va bien,
— 301 (redirection permanente, permettant de changer la méthode HTTP), 302 (redirection temporaire, permettant de changer la méthode), 307 (redirection temporaire, ne permettant pas de changer la méthode), 308 (redirection permanente, ne permettant pas de changer la méthode) : les divers codes de redirection, permettant à un « méta-serveur » d'indiquer le bon serveur RDAP dans l'en-tête `Location:` de la réponse (par exemple, en théorie, l'IANA pourrait avoir un serveur RDAP qui redirige vers les serveurs RDAP des TLD),
— 404 : ce truc n'a pas été trouvé, par exemple parce que je cherche un nom de domaine qui n'existe pas,
— 400 : requête mal formée, parce que le client RDAP est bogué ou bien parce qu'on a fabriqué une requête RDAP à la main, en se trompant,
— 429 : on envoie trop de requêtes (ce code a été normalisé dans le RFC 6585). Aujourd'hui, les serveurs whois ont souvent ce problème de requêtes répétées, cela peut être une attaque DoS mais aussi un domaineur qui veut récolter de l'information, ou un autre type de malhonnête qui veut se constituer une base de données. Un serveur RDAP aura donc probablement une fonction de limitation du trafic, et répondra 429 si le client exagère. Les clients honnêtes ralentiront alors.

Voici quelques exemples réels avec le serveur RDAP expérimental d'APNIC et le client HTTP curl. D'abord, un cas où tout se passe bien :

```
% curl -v http://rdap.apnic.net/ip/2001:dc0:2001:11::194
> GET /ip/2001:dc0:2001:11::194 HTTP/1.1
> User-Agent: curl/7.26.0
> Host: rdap.apnic.net
> Accept: */*
>
< HTTP/1.1 200 OK
< Date: Thu, 05 Mar 2015 16:08:37 GMT
< Content-Type: application/rdap+json
< Access-Control-Allow-Origin: *
< Transfer-Encoding: chunked
< Server: Jetty(9.2.z-SNAPSHOT)
```

Un cas où l'objet demandé n'existe pas et où le serveur renvoie le fameux code HTTP 404 :

```
% curl -v http://rdap.apnic.net/ip/4001:dc0:2001:11::194
> GET /ip/4001:dc0:2001:11::194 HTTP/1.1
> User-Agent: curl/7.26.0
> Host: rdap.apnic.net
> Accept: */*
>
< HTTP/1.1 404 Not Found
```

Et enfin un cas où l'objet existe chez un autre RIR et on est donc renvoyé au RIPE-NCC, avec le code normal de redirection HTTP :

```
% curl -v http://rdap.apnic.net/ip/2001:4b98:dc0:41::
> GET /ip/2001:4b98:dc0:41:: HTTP/1.1
> User-Agent: curl/7.26.0
> Host: rdap.apnic.net
> Accept: */*
>
< HTTP/1.1 301 Moved Permanently
< Date: Thu, 05 Mar 2015 16:11:16 GMT
< Location: http://rdap.db.ripe.net/ip/2001:4b98:dc0:41::
```